AMENDMENTS TO THE CLAIMS

Please amend the claims as indicated in the following listing of all claims:

1. – 25. (Cancelled)

26. (New) A method for handling sharing of a physical memory space between a first process and a second process, the method comprising:

allocating a first address range in the first process and a second address range in the second process, wherein one of the processes executes native code of a program and the other process executes safe language code of the program; and

mapping the first address range and the second address range to the shared physical memory space.

27. (New) The method of claim 26, wherein at least one of the first and second address ranges comprise virtual addresses.

28. (New) The method of claim 26, wherein the physical memory spaces comprises a set of one or more physical pages.

29. (New) The method of claim 26, wherein the allocating is responsive to the first process requesting a direct buffer for the first address range from the second process.

30. (New) The method of claim 29 further comprising the second process:

causing generation of a direct buffer object and an associated direct buffer object identifier; and

communicating the direct buffer object identifier and a physical memory space identifier for the shared physical memory space to the first process.

31. (New) The method of claim 29 further comprising communicating the direct buffer object identifier and the physical memory space identifier to other processes that execute native code.

32. (New) The method of claim 26 further comprising maintaining an encoding that indicates at least one of overlapping address ranges and nested address ranges within one of the first and second processes.

33. (New) A method for sharing one or more physical memory spaces between a first and second process, the method comprising:

maintaining analogous memory address ranges between the first and second processes,

wherein one of the first and second processes executes native code and the other process executes safe language code,

wherein the address ranges in the first process and the analogous address ranges in the second process are mapped to same portions of the one or more physical memory spaces.

34. (New) The method of claim 33 wherein the maintaining comprises:

the first process reserving a first address range and requesting a buffer from the second process;

responsive to the request from the first process, the second process allocating a second address range analogous to the first address range.

35. (New) The method of claim 34 further comprising:

the second process creating a buffer object and mapping the second address range to a first portion of the one or more physical memory spaces and communicating a buffer object identifier and a physical memory space identifier that identifies the first physical memory space portion to the first process;

the first process mapping the first address range to the first physical memory space portion as identified by the physical memory space identifier.

36. (New) The method of claim 35 further comprising the first process communicating the buffer object identifier to a native code caller.

37. (New) The method of claim 33 wherein the physical address ranges comprise a set of one or more physical memory pages.

38. (New) The method of claim 33 further comprising allowing at least two of the virtual address ranges in the first process to overlap.

39. (New) The method of claim 38 further comprising:

the first process requesting an address range [A1, A1+S1] that overlaps with a previously allocated address range [A2, A2+S2], wherein A1 and A2 represent addresses in the first process and S1 and S2 represent memory space sizes;

the second process,

allocating an address range [A2', A2'+S2], wherein A1' and A2' represent addresses in the second process,

mapping [A2', A2' + (A1+S1-A2)] in the second process to a same first portion of the one or more physical memory spaces to which [A2, A1+S1] in the first process is mapped, and

mapping [A2'+(A1+S1-A2), A2'+S2] in the second process to a same second portion of the one or more physical memory spaces to which [A1+S1, A2+S2] in the first process is mapped.

40. (New) The method of claim 33 further comprising maintaining a list that indicates the address ranges, wherein the list allows detection of at least one of overlapping address ranges and nested address ranges.

41. (New) The method of claim 33, wherein at least one of the first process address ranges and the second process address ranges comprise virtual addresses.

42. (New) A computer program encoded on one or more computer readable media, the computer program comprising:

a first language code executable to allocate an address range in a first environment, which executes the first language code, responsive to a request for a buffer, and executable to map the first environment address range to a physical memory space; and

a second language code executable to request the buffer and allocate an address range in a second environment, which executes the second language code, and executable to map the second environment address range to the physical memory space responsive to indication of the physical memory space from the first language code,

wherein the first environment address range is correspondent to the second environment address range,

wherein one of the first and second language codes comprises a safe language code and the other of the first and second language codes comprises a native code.

43. (New) The computer program encoding of claim 42 further comprising an interface code to handle communications between the first and second environments.

44. (New) The computer program encoding of claim 43, wherein the interface code is implemented according to the Java native interface, the safe language code is implemented according to the Java programming language, and the one of the first and second environments that executes the safe language code comprises a virtual machine.

45. (New) The computer program encoding of claim 44, wherein the other of the first and second environments that executes native code comprises an operating system.

46. (New) The computer program encoding of claim 42, wherein the first environment address range being correspondent to the second environment rage comprises the address ranges being a same size.

47. An apparatus comprising:

a memory; and

means for allocating an address range in a first process for a first code and an analogous address range in a second process for a second code to facilitate sharing of at least a portion of the memory between the first code and the second code,

wherein one of the first and second processes executes native code and the other of the first and second processes executes safe language code.

48. The apparatus of claim 47, wherein at least one of the first process address range and the second process address range comprises virtual addresses.

49. The apparatus of claim 47 further comprising means for allowing detection of at least one of nested address ranges and overlapping address ranges.